KIP-457: Add DISCONNECTED status to Kafka Streams

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *KAFKA-6520*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

When a Kafka Streams application is started, and then is unable to connect (docker daemon for instance is killed), then the user does not have an easy way of identifying the current state of Kafka Streams (i.e. does not know that its DISCONNECTED). For users who wish to know the status of their application, they would have to use alternative means of performing a health check. This KIP works to resolve this issue by adding a method to check for connection.

Public Interfaces

In Kafka Streams, a method KafkaStreams#isConnected() will be added. When something happens unexpectedly which causes the connection to vanish, `KafkaStreams#isConnected()` will return false. Please note that the difference between `DISCONNECTED` and `DEAD` is that KafkaStreams, when it is in its dead state, is no longer running. While in the `DISCONNECTED` case, it would still be alive, but could not connect to broker. We would add a new state to `getState()` so that it will return `State.DISCONNECTED` should the consumer be disconnected to the broker.

The enum `State` found in KafkaStreams would be modified as follows:

public enum State

```
/**
 * Stream thread states are the possible states that a stream thread can be in.
 * A thread must only be in one state at a time
 * The expected state transitions with the following defined states is:
 *
 * <pre>
 *                +-------------+
 *          +<--- | Created (0) |
 *          |     +-----+-------+
 *          |           |
 *          |           v
 *          |     +-----+-------+
 *          +<--- | Starting (1)|<-----------+
 *          |     +-----+-------+            |
 *          |           |                    v
 *          |           |            +---------------+
 *          |           v            |DISCONNECTED(7)|
 *          |     +-----+-------+    +---------------+
 *          +<--- | Partitions  |      ^    ^    ^
 *          |     | Revoked (2) |      |    |    |
 *          |     +-----+-------+<-----+    |    |
 *          |           |                    |    |
 *          |           v                    |    |
 *          |     +-----+-------+            |    |
 *          |     | Partitions  |            v    |
 *          +<--- | Assigned (3)| <--------->+    |
 *          |     +-----+-------+                 |
 *          |           |                         |
 *          |           v                         |
 *          |     +-----+-------+                 v
```

```
     *          |     | Running (4) | <------------->+
     *          |     +-----+-------+
     *          |           |
     *          |           v
     *          |     +-----+-------+
     *          +---> | Pending     |
     *                | Shutdown (5)|
     *                +-----+-------+
     *                      |
     *                      v
     *                +-----+-------+
     *                | Dead (6)    |
     *                +-------------+
     * </pre>
    *
     * Note the following:
     * <ul>
     *     <li>Any state can go to PENDING_SHUTDOWN. That is because streams can be closed at any time.</li>
     *     <li>
     *         State PENDING_SHUTDOWN may want to transit to some other states other than DEAD,
     *         in the corner case when the shutdown is triggered while the thread is still in the rebalance
loop.
     *         In this case we will forbid the transition but will not treat as an error.
     *     </li>
     *     <li>
     *         State PARTITIONS_REVOKED may want transit to itself indefinitely, in the corner case when
     *         the coordinator repeatedly fails in-between revoking partitions and assigning new partitions.
     *         Also during streams instance start up PARTITIONS_REVOKED may want to transit to itself as well.
     *         In this case we will forbid the transition but will not treat as an error.
     *     </li>
     *     <li>
     *         Any state which is considered running (i.e. STARTING, RUNNING, PARTITIONS_REVOKED,
PARTITIONS_ASSIGNED) could transition
     *         to and from DISCONNECTED status. This state indicates that the StreamThread is alive and well,
but the connection to
     *         broker does not exist.
     *     </li>
     * </ul>
     */
    public enum State implements ThreadStateTransitionValidator {
        CREATED(1, 5), STARTING(2, 5, 6), PARTITIONS_REVOKED(3, 5, 6), PARTITIONS_ASSIGNED(2, 4, 5, 6), RUNNING
(2, 5, 6), PENDING_SHUTDOWN(6), DISCONNECTED(2,3,4,5), DEAD;

        private final Set<Integer> validTransitions = new HashSet<>();

        State(final Integer... validTransitions) {
            this.validTransitions.addAll(Arrays.asList(validTransitions));
        }

        public boolean isRunning() {
            return equals(RUNNING) || equals(STARTING) || equals(PARTITIONS_REVOKED) || equals
(PARTITIONS_ASSIGNED) || equals(DISCONNECTED);
        }

        @Override
        public boolean isValidTransition(final ThreadStateTransitionValidator newState) {
            final State tmpState = (State) newState;
            return validTransitions.contains(tmpState.ordinal());
        }
    }
```

This would also mean that a new method would be added to KafkaConsumer to allow the StreamThread to query the health of the connection.

KafkaConsumer#isConnected()

```
/**
 *        @return whether or not the connection is alive
 */
public boolean isConnected();
```

Proposed Changes

We would query individual StreamThreads for their individual status and update the state accordingly.

Compatibility, Deprecation, and Migration Plan

This would not have any compatibility issues with previous versions. Changers are internalized and since the version of messages are not a concern, no upgrade path should be necessary.